
# A probe into process-level attack detection in industrial environments from a side-channel perspective

# A Probe into Process-Level Attack Detection in Industrial Environments from a Side-Channel Perspective

Wissam Aoudi
Chalmers University of Technology
Gothenburg, Sweden
wissam.aoudi@chalmers.se

Albin Hellqvist
Chalmers University of Technology
Gothenburg, Sweden
albin.hellqvist@gmail.com

Albert Overland
Chalmers University of Technology
Gothenburg, Sweden
albert.overland@gmail.com

Magnus Almgren
Chalmers University of Technology
Gothenburg, Sweden
magnus.almgren@chalmers.se

## Abstract

Process-level detection of cyberattacks on industrial control systems pertain to observing the physical process to detect implausible behavior. State-of-the-art techniques identify a baseline of the normal process behavior from historical measurements and then monitor the system operation in real time to detect deviations from the baseline. Evidently, these techniques are intended to be connected to the control flow to be able to acquire and analyze the necessary measurement data, which makes them susceptible to compromise by the attacker. In this paper, we approach process-level attack detection from a side-channel perspective, where we investigate the feasibility and efficacy of monitoring industrial machines through external sensors. The sensors measure physical properties of the process that are bound to change during a cyberattack. We demonstrate the viability of our approach through simulations and experiments on real industrial machines.

## Keywords

Anomaly Detection; PASAD; Embedded System; Industrial Environment; Industrial Control System

**ACM Reference Format:**
Wissam Aoudi, Albin Hellqvist, Albert Overland, and Magnus Almgren. 2019. A Probe into Process-Level Attack Detection in Industrial Environments from a Side-Channel Perspective. In *Fifth Annual Industrial Control System Security (ICSS) Workshop (ICSS), December 10, 2019, San Juan, PR, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3372318.3372320


*ICSS, December 10, 2019, San Juan, PR, USA*

ACM ISBN 978-1-4503-7719-5/19/12…$15.00
https://doi.org/10.1145/3372318.3372320

## 1 Introduction

The benefits of connecting Industrial Control Systems (ICS) that control physical, and often safety-critical, processes to the Internet and the satisfaction it offers to industrial stakeholders suggest that the trend of digitalization and connectivity in this domain will continue to expand. Inevitably, the reality to be foreseen is one where malicious actors with various motives leverage this crossover from the cyber world to the mechanical world to launch sophisticated cyberattacks that have potential to cause unsustainable losses to critical infrastructure. Conventional IT-based security, such as encryption, firewalls, access control and so forth may, with proper adjustments, be applicable to parts of these systems. However, in light of the increasingly reported successful cyberattacks on vital infrastructure, the need for more advanced and tailored solutions that can guard the physical-level operation has been pressing over the recent years. In this context, techniques that directly monitor industrial processes through process data have been one of the driving efforts for achieving an additional layer of security in ICS [3–5, 7, 9–11]. In particular, model-free techniques, which stem from different disciplines including statistics, machine learning, and time-series analysis, do not require a specification of the physical process. They share the common purpose of establishing a baseline for the benign behavior of the system from historical measurements, and then detecting deviations from the baseline in real time.

In this work, we explore the viability of using the mentioned techniques for side-channel based monitoring of industrial machinery. The principal idea is that industrial machines are poised to exhibit changes in physical properties, such as vibration and sound, when an attack is undergoing. As these properties can be measured with sensors, process-level attack-detection mechanisms may be used to detect such changes in behavior.

Our side-channel based approach has the following merits: *i)* the detection system is relatively cheap and practical to deploy; *ii)* it is completely isolated, hence unreachable by the attacker; *iii)* and it makes fewer assumptions about data collection since it generates its own data.

Aoudi et al. [3] and Almgren et al. [2] have demonstrated that process-level intrusion detection systems can accurately detect stealthy attacks and run reliably in real industrial settings. However, the way their proposed system is deployed shows high dependence on the monitored process. This work, on the other hand, shows that it is not only feasible, but also practical and cost-efficient, to deploy a duplicate system for more robust monitoring. Other related works in the literature have focused on fingerprinting to authenticate sensors and on performing side-channel analysis to detect changes in software behavior. For instance, Ahmed et al. [1] use noise patterns in sensor measurements, which appear due to manufacturing imperfections, to detect data integrity attacks. Van Aubel et al. [12] propose to use electromagnetic measurements to detect behavioral changes in ICS software. Our proposed detection approach differs from similar hardware-based analysis techniques in that it works directly at the machine level. We tested our technique on an industrial metal lathe and a drilling machine and managed to successfully detect realistic attacks on them.

We present our methodology in Section 2, and lay down principles of signal processing in Section 3. We state the design choices we made in Section 4 and how these choices are realized in Section 5. Finally, we evaluate our approach in Section 6 and conclude this work in Section 7.

## 2 Methodology

Three main components make up our system: the sensors, the attack-detection algorithm, and the embedded system on which the detection algorithm runs. The sensors measure the desired physical properties of the machines. The generated measurements are then preprocessed and fed into the detection algorithm, which computes a score for every measurement to determine if the process behavior is drifting from the baseline behavior.

We chose PASAD, a process-aware stealthy-attack detection mechanism recently proposed by Aoudi et al. [3], as the detection algorithm running on the embedded system. The choice of this algorithm as well as the choices for the sensors and the embedded system are motivated in Section 4.

Initially, PASAD learns the normal behavior from historical sensor measurements in an offline training phase. The learning is achieved by embedding the time series of measurements in a vector space and then performing a *spectral decomposition* of a matrix made up of the training vectors to identify a *signal subspace*, where the deviation from normal behavior is more detectable. More specifically, when mapped to the signal subspace, the sensor measurements form a cluster during normal system operation, and depart from the said cluster when the system is experiencing changes in behavior.

The parameters from the the training phase are then used in an online detection phase, which is the part of the algorithm that we implement on the embedded system to detect attacks in real time. The detection works by computing a so-called *departure score* for every incoming sensor measurement to determine whether the physical process is under attack. The elements of the detection phase are explained in more detail in Section 5.

## 3 Background

The detection algorithm runs in the digital domain, whereas the measured signals fed into the algorithm are analog in nature; this necessitates a kind of conversion of input signals from analog to digital. This section lays down concepts related to analogue-signal handling and well-established techniques in electrical engineering that helped us build the desired prototype and achieve our design goals, such as cheap hardware and independence of the actual process sensors.

The conversion from the analog domain to the digital domain is achieved using an Analog-to-Digital Converter (ADC). The input signal to an ADC consists of a voltage that needs to be within a voltage range defined by the ADC's specification. The ADC converts the voltage at the input to a digital value where the range of this value is determined by the ADC's number of bits $N$ such that $N$ bits allow $2^N - 1$ quantization levels to be represented. For example, a 12-bit ADC with a full scale voltage range of 3.3 V has a voltage resolution of approximately 806 µV. This translates into one of the 4096 steps available for a 12-bit ADC, equivalent to 806 µV. In order to take advantage of all quantization levels provided by the ADC, it is recommended to make the analog input signal span the entire full scale voltage range. However, it is common for analog signals from sensors to have amplitudes many times smaller than the ADC's full scale voltage range. This can be rectified by amplifying signals with small amplitudes using an operational amplifier circuit in a non-inverting configuration with two resistors as shown in Fig. 1a.

The output of an analog sensor often spans both positive and negative voltages. Since most ADCs use unipolar voltages, it is necessary to convert the negative voltages to positive voltages by introducing an offset. Restricting a voltage range to only positive voltages can be performed by using an operational amplifier summing circuit, shown in Fig. 1b. This type of circuit allows a direct voltage offset (bias) to be added on top of the input signal, resulting in the addition of the two. Knowledge of the maximum expected negative value on the input can be used to derive a sufficient DC bias to ensure that the output signal will not, in theory, become negative.

If the expected frequency content of the measured signal is known, it is sensible to suppress signals with frequencies outside of the chosen spectrum to minimize the out-of-band noise. One way to suppress the unwanted high-frequency components is to pass the signal through a low-pass filter with a cut-off frequency determining where the attenuation of the filter starts to have a significant impact on the signal. Usually, the cut-off frequency is defined as the frequency where the attenuation of the signal is −3.01 dB. A simple way to construct a low-pass filter is using a resistor and a capacitor as shown in Fig. 1c, which depicts a first-order filter, meaning that it only consists of one frequency-dependent element. As a consequence, the attenuation roll-off rate in

the stop band is limited to $-20\,\mathrm{dB}$/decade. Higher-order filters provide steeper roll-off at the expense of needing more components. In a similar fashion, one could construct a high-pass filter by switching the places of the resistor and the capacitor.

Some sensors have a significant capacitance associated with them, which needs to be factored in during implementation. Essentially, depending on the capacitance of the sensor being used, as well as how low frequencies are to be measured, significantly high resistance values may be needed in order to reduce the filtering effects of the sensor's capacitance. If such high resistance values are not available when building the measurement circuit, one can use a bootstrap circuit, such as the one depicted in Fig. 1d, to emulate the needed resistance using resistance values that are orders of magnitude lower. It is recommended, however, to use a high resistance value since it decreases the complexity of the design and is most likely cheaper to construct in addition to a lower power draw. The capacitor $C_1$ will prevent any direct current to impact the positive feedback of the circuit, meaning it will only feed back signals that vary in time, such as sine waves. The resistors $R_3$ and $R_2$ form a voltage divider from the output of the circuit, which results in a voltage potential at point A.

Finally, it is worth noting that any system that converts analog signals to a discrete representation through sampling needs to fulfill the Nyquist–Shannon sampling theorem, which states that the minimum sampling rate of the system needs to be higher than twice the bandwidth of the sampled signal in order to avoid aliasing.

## 4 System Design

One of our main goals in this work is to build a detection prototype that is cheap, practical to deploy, and independent of the system being monitored. The system consists of sensors that measure relevant physical properties of the industrial machines, an STM32F767ZI microcontroller as an embedded system running the attack detection algorithm, and a PC. This section states and motivates the design choices we made pertaining to the types of sensors employed, the anomaly-detection algorithm, and the embedded system. An overview of the system can be seen in Fig. 2.

### 4.1 Choice of Sensors

Three types of off-the-shelf sensors were decided to be used: a microphone sensor, a load sensor, and a vibration sensor.

The **microphone sensor** allows for controlled tests even in presence of some background noise. By placing the microphone in a quiet setting, strict control of noise or disturbances can be achieved. Furthermore, complex test cases can be emulated by playing multiple sound sources to the microphone at the same time. As a result, the microphone sensor allows easy testing of a variety of wave forms with different frequencies and shapes. The output of the microphone sensor circuit can either be a sine wave from the microphone or a DC voltage corresponding to the frequency of the sine wave's output using a frequency-to-voltage converter. Using both microphone sensor and frequency sensor circuits widens the input frequency range of the signal without violating the Nyquist-Shannon sampling theorem. Moreover, this sensor property enables two types of testing scenarios with the voltage type requiring less resources due to the fact that the minimum required sampling rate of the ADC does not depend on the actual frequency of the microphone's sine wave since the latter is transformed to a DC voltage. However, the trade-off is that we lose all information about the waves' amplitudes.

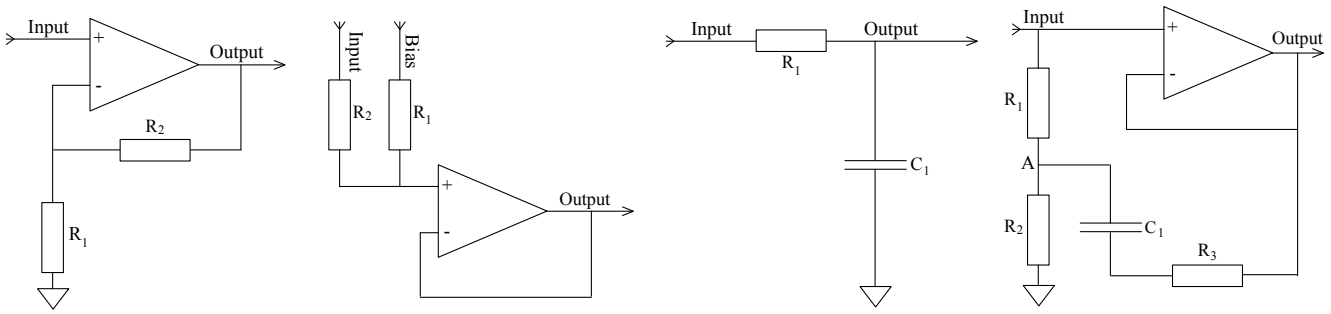
The **load sensor** provides a simple case as it produces a constant output provided that the load on the sensor remains constant. While the microphone sensor produces constantly changing output values, the load sensor is more stable and only changes its output when the load changes. This makes the load sensor suitable as a reference case, where barely any other factors than what it is loaded with come into play. In particular, background noise can influence measurements in the case of the microphone sensor and slightly distort the output, and vibrations from adjacent machinery could also affect the measurements of the vibration sensor. The load sensor uses a strain gauge load cell with a capacity of $10\,\mathrm{kg}$. Choosing a relatively low maximum load value allows for easy prototyping and testing. The resistance of strain gauge load cell changes roughly linearly with the load. This change in resistance can be detected and translated into a voltage that the ADC on the microcontroller can interpret.

The **vibration sensor** is inherently robust against various noise sources when mounted on machinery in realistic environments. Similar to the microphone sensor, the output of the vibration sensor is a sine wave, which is subject to sampling rate requirements. However, the vibration sensor used in our work is designed for low frequencies bounded below $40\,\mathrm{Hz}$, which severely reduces the sampling requirements compared to sound waves in the audible range. The vibrations generated from machines, such as lathes or fans, often vary in intensity and are more difficult to control than, for example, the volume of a speaker. Consequently, to allow measuring different vibration scenarios with this sensor, it is sensible to have a variable gain in the vibration sensor circuit in order to match the signal's output amplitude of the sensor circuit to the full-scale voltage range of the ADC.

### 4.2 Choice of Detection Algorithm

As stated in Section 2, the anomaly-detection algorithm of choice in this work is PASAD. For the detection algorithm to fit our approach, it is required to be model-free and capable of detecting changes in the process behavior by solely processing time series of sensor measurements. There exist other candidates in the literature, such as autoregressive models and neural network-based methods, but they lack important features that make PASAD better suited for our purposes.

Compared to PASAD and neural networks, the main operation of the autoregression method in the detection phase is simple and lightweight as it mainly consists of a prediction step based on a relatively simple linear combination of a




**(a) Non-inverting amplifier circuit** **(b) Amplifier summing circuit** **(c) First-order low-pass filter** **(d) Resistance bootstrap circuit**

**Figure 1: Circuitry used for handling the analogue sensor signals.**

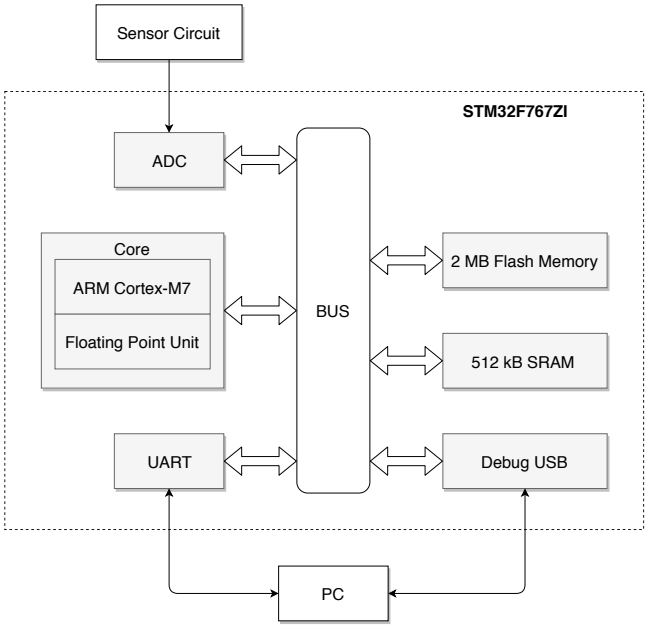



**Figure 2: System overview where the STM32F767ZI is within the dotted lines.**

number of recent values. However, this simplicity comes at the expense of low detection accuracy in presence of noise [3].

The neural network-based method proposed in [9], where the network architecture is determined by a genetic algorithm, is claimed to have potential to become quite efficient at detecting anomalies in a time series of data. However, this hinges upon the algorithm being given extensive time for training in addition to having formidable hardware resources available. Even with access to powerful hardware, the time it takes to come up with the optimal architecture still takes a long time.

PASAD exhibits better detection rates and is capable of detecting more types of attacks than the autoregressive model [3]. One of the reasons for the better performance is that the algorithm is highly insensitive to noise, which makes it better at detecting attacks that are stealthy in nature, wherein attackers try to hide the attacks within the noise level. However, PASAD does use more resources than the autoregression method for its detection phase, especially when it comes to memory usage. The additional memory used is due to the higher storage size of the algorithm's output from the training phase. However, the memory requirement does not seem to be absurdly high in comparison to what is available on many commercial microcontrollers. The scenario with the highest memory usage presented in [3] uses around 1.7 MB of data which is a reasonable size of the flash memory available on a microcontroller.

### 4.3 Choice of Embedded System

The choice of the STM32F767ZI microcontroller as an embedded system (see Fig. 2) for running PASAD was made after a comparison with two other potential candidates: Raspberry Pi 1 Model A+ and Arduino Due. The metrics used in the comparison were computational performance, amount of memory, the availability of analog-to-digital conversion, and double-precision floating-point support.

When it comes to performance, the Raspberry Pi is considerably faster with its 700 MHz clock frequency compared to the Arduino Due's 84 MHz and the STM32F767ZI's 216 MHz. However, the Raspberry PI's RAM is of SDRAM type that has a slower access time than the SRAM, which the other microcontrollers feature. On the other hand, the other microcontrollers do not have enough RAM to store all the training vectors, which necessitates real-time transfer of data between the RAM and the flash memory.

The Arduino Due does not have hardware support for floating-point calculations, which forces the system to perform these operations in software instead of hardware, thereby significantly reducing the performance of its floating-point operations.

In terms of memory, the Raspberry Pi outperforms the other microcontrollers because of its large RAM, which makes it possible to store all the training vectors without having to load parts of them from any non-volatile memory in real time. The STM32F767ZI's RAM can hold all of the training vectors

in most of the experimental cases. In the other cases, the microcontroller's clock frequency and transfer speed from the flash memory makes it feasible to temporarily transfer the necessary vectors from the flash memory to the RAM in real time. Implementing a real-time memory transfer for the Arduino Due would be complex due to its limited RAM size, which is further complicated by the lack of enough non-volatile memory to store training vectors for the more demanding experiments.

The Arduino Due and the STM32F767ZI have built-in ADCs, all of them supporting a resolution of 12 bits. These ADCs are considered to be adequate when it comes to the sampling rate and resolution requirements of commonly used sensors. However, the Raspberry PI does not feature any built-in ADC and therefore needs an external ADC connected to it, which would add to the complexity of the embedded system.

Since the task was to find an adequate resource-constrained embedded system, the optimal choice was decided to be the STM32F767ZI microcontroller due to its reasonably fast CPU, hardware support for double-precision floating-point format, fast and sufficient memory, built-in ADC, and low power consumption.

## 5 Implementation

Several challenges arise when attempting to execute the top-level design decisions described in Section 4 in terms of both hardware and software implementation. One of the major challenges is to implement PASAD on such a severely limited hardware as a microcontroller. This section details such an implementation, where the supporting sensor circuitry needed to enable the measurement of analog signals and preparing these for the microcontroller is presented, followed by a description of the software implementation of the anomaly-detection system on the microcontroller.

### 5.1 Interfacing the Microcontroller and the PC

The STM32F767ZI microcontroller features a debug USB by default that can be used to flash the code of the program. However, since it is a debug USB, it cannot be used to transfer any non-debug data and commands between the microcontroller and the PC in real time. To work around this, we chose to use the UART interface, which is supported by many systems and can be used to control the microcontroller from a machine with a simpler interface, such as a PC. The asynchronous serial communication between the microcontroller and the PC was set to have a speed of 115 200 baud per seconds since this was thought to be the highest stable speed that could be used. This speed corresponds to a data rate of 11 520 bytes per second.

In addition to the hardware interfacing, we developed a program that runs on the PC to meet the needs of the microcontroller. The program was programmed to have four functionalities: *i)* Send a command to the microcontroller to start collecting values that can be used in the training phase; *ii)* Create a `.csv` file from the binary file extracted from the flash memory that contains the collected values to be used in the training phase; *iii)* Create a binary file from training vectors that can be used when flashing the microcontroller's flash memory; and *iv)* Send a command instructing the microcontroller to start running the PASAD algorithm.

### 5.2 Interfacing the Microphone

The microphone sensor is an electret FC-109 MAX9812 Microphone Amplifier Module that outputs both positive and negative voltages approximately in the range $\pm 35\,\text{mV}$ at sound pressure levels of $\approx 60\,\text{dB}$. Since the ADC is designed to accurately convert voltages within 0 V and 3.3 V, the interfacing circuit should raise the negative voltages above 0 V, which is achieved using a unipolar-to-bipolar conversion. Since it is recommended to use the entire available range of the ADC in order to use as many bits as possible for the quantized signal, the interfacing circuit should also amplify the signal level to span the entire available voltage range 0 V to 3.3 V. Both of the operations can be performed using operational amplifiers in combination with a few resistors. Raising the voltage can be done using a summing circuit and amplifying the signal can be performed using a non-inverting operational amplifier, where the values of the supporting resistors connected to the operational amplifier determine the gain (see Section 3). Next, the signal is filtered by a simple resistor-capacitor circuit to keep the signal of interest, which is in the human hearing range ($< 20\,\text{kHz}$).

In order for the signal to be ready for the ADC, the input voltages need to remain within the 0 V to 3.3 V range. Since the output of the microphone depends on the input sound level, sounds that are well above the conversation sound level will result in signals with higher amplitudes than what the system is designed for. To avoid this situation, a pair of Schottky diodes are placed after the filter connected to the 3.3 V source and ground respectively.

With a working microphone circuit, the output of this circuit is ready to be sent to the ADC. However, the microphone circuit has also been extended by introducing another circuit that converts the frequency of the output signal to a constant voltage, i.e., a DC signal that can be fed to the ADC. This allows the microphone circuit to work in two modes, either by sending varying signals to the ADC that takes into account both its amplitude and frequency, or by sending a DC signal to the ADC where the amplitude is omitted.

### 5.3 Interfacing the Load Sensor

The second sensor consists of a load cell fastened between two plates that function as the surface on which weights can be placed. A load cell is a weight measurement device consisting of a metal beam that bends when force is applied to it. The cell has a strain gauge that is connected to the metal beam of the load cell, which bends along with the metal beam when force is applied to it. When the strain gauge is bent, the change in its resistance can be measured to derive how much force has been applied to the load cell. However, the change in resistance is oftentimes extremely small, which makes it difficult to measure. To allow such small changes in

resistance, the strain gauge is connected to one of the legs in a Wheatstone bridge circuit, which allows exceptionally precise measurement of the strain gauge's resistance [6].

Even though the Wheatstone bridge allows the detection of small changes in resistance, the resulting voltage from the Wheatstone bridge will still be extremely small and needs to be amplified [8]. An instrumentation amplifier can be used for this task since it has a high input impedance, thus limiting the effect it has on the input circuit. The instrumentation amplifier used to interface the output of the load cell is INA217, which has an input impedance of 60 MΩ. The gain of the INA217 is determined by the size of the resistor placed between the instrumentation amplifier's inputs.

## 5.4 Interfacing the Vibration Sensor

The vibration sensor is the capacitive piezoelectric sensor MiniSense 100, which can detect vibrations up to 40 Hz with a sensitivity of 1.1 Vg, where g is the gravitational acceleration. The output of the sensor ranges between ±90 V depending on how powerful the vibrations are. However, providing vibrations that actually produce this high voltage requires an immense acceleration of over 80 g in either direction. As such, the output is usually rather small and needs amplification. However, the amplification could result in the voltage exceeding the intended range. To alleviate this effect, additional over-voltage protection diodes are placed at the sensor output. Moreover, the capacitive sensor creates a high-pass filter with the resistance of the circuit between the sensor and the ADC. Since the capacitance of the MiniSense 100 is 244 pF, the external resistance needs to be sufficiently large to avoid attenuating the low-frequency signals.

When taking measurements with the vibration sensor it is often a practical necessity to use wires to extend the physical range of the sensor. The wires will introduce some capacitance, and as a result slightly change the characteristics of the high-pass filter formed from the sensor capacitance and the bootstrap circuit. This can give rise to peaks in amplitude at the lowest frequencies below 1 Hz. To remedy this, a high-pass filter is placed after the bootstrap circuit with a cut-off frequency of approximately 0.72 Hz. Following the high-pass filter is an amplification stage, a bipolar-to-unipolar conversion circuit, a low-pass filter and over-voltage protection for the ADC similar to the microphone circuit. The low-pass filter is designed to have a cut-off frequency of approximately 34 Hz, which is within the specified upper limit of 40 Hz according to the vibration sensor specification.

## 5.5 Implementation

This section describes how the implementation of PASAD on the STM32F767ZI was performed. First, we explain how the ADC was set up in order to have a more accurate sampling rate. Then, the training phase and the detection phase of the algorithm are described, which both use the ADC to collect values. The collection of values by the ADC is performed using interrupts, i.e., running concurrently together with other software.

*5.5.1 Timer-based Analog-to-Digital Conversion.* The ADC on the chosen STM32F767ZI microcontroller is rather limited when it comes to setting the desired sampling rate. Not only are there many parameters that can be changed, but it would also be difficult to achieve a sampling rate close to the desired sampling rate. In order to have a more accurate sampling rate that also is easier to adjust, a timer peripheral that decides when the ADC should sample is used. The timer peripheral runs at 54 MHz, and the adjustment of the sampling rate works by setting the timer period to a specific value. The timer works by counting up at each clock tick until it reaches the specified timer period, at which time, it sends a signal that instructs the ADC to sample one value.

*5.5.2 PASAD's Training Phase.* PASAD learns the deterministic behavior of the signal underlying the time series of measurements during a training phase. As mentioned in Section 2, training of the algorithm is an offline procedure, which was performed on a PC using a MATLAB script. The training parameters are a number $N$ of initial sensor measurements for training, a lag parameter $L$ as a sliding-window length, and a so-called statistical dimension $r$ indicating the number of eigenvectors that sufficiently describe the deterministic signal. The output of this phase is a set of training vectors and a centroid $c$.

After various mathematical operations have been run on the subseries of $N$ sensor measurements, an $L$-by-$L$ matrix consisting of $L$ eigenvectors is produced. Also, the result of taking the $r$ leading eigenvectors from the $L$-by-$L$ matrix produces an $L$-by-$r$ matrix denoted as $U$. The $U$ matrix is then transposed into an $r$-by-$L$ matrix $U^{\mathrm{T}}$, which is subsequently used in the detection phase to project vectors onto the $r$-dimensional linear subspace $\mathcal{L}^r$ spanned by the $r$ leading eigenvectors.

In addition to the matrix $U^{\mathrm{T}}$, the $c$ vector is transformed into a $\tilde{c}$ vector of $r$ elements representing the centroid of the cluster of training vectors in the signal subspace $\mathcal{L}^r$. Finally, an additional $w$ vector of $r$ elements is initialized to contain the weight distribution of the $r$ leading eigenvalues for computing the weigthed Euclidean distance in the detection phase.

*5.5.3 PASAD's Detection Phase.* In the detection phase, we initialize a lagged vector $x$ to contain the $L$ latest sensor measurements. The detection phase is built around projecting the $x$ vector onto the signal subspace $\mathcal{L}^r$ and then computing its distance from the centroid $\tilde{c}$. This is mathematically done by calculating the squared Euclidean distance between the projected $U^{\mathrm{T}}x$ vector and the centroid $\tilde{c}$ as

$$D = ||w \circ \tilde{c} - U^{\mathrm{T}}x||^2, \tag{1}$$

where $\circ$ denotes the Hadamard (element-wise) product. To simplify Equation 1, we break it down into three steps. First, the $x$ vector is projected onto the signal subspace $\mathcal{L}^r$ by computing $p = U^{\mathrm{T}}x$

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_r \end{bmatrix} = \begin{bmatrix} U_{1,1} & U_{2,1} & \cdots & U_{L,1} \\ U_{1,2} & U_{2,2} & \cdots & U_{L,2} \\ \vdots & \vdots & \ddots & \vdots \\ U_{1,r} & U_{2,r} & \cdots & U_{L,r} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_L \end{bmatrix}. \quad (2)$$

In the second step, the projected vector is compared to the centroid $\tilde{c}$, which produces a temporary vector denoted as $y$ such that

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} = \begin{bmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ \vdots \\ \tilde{c}_r \end{bmatrix} - \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_r \end{bmatrix}. \quad (3)$$

**Algorithm 1** Initialization of the PASAD algorithm.

```
1:  r ← Mem Base_addr
2:  L ← Mem Base_addr + 4
3:
4:  c_addr ← Base_addr + 16
5:  w_addr ← c_addr + r · 8
6:  U^T_addr ← w_addr + r · 8
7:
8:  for i = 0 to r − 1 do
9:      c_i ← Mem c_addr + i · 8
10: end for
11:
12: for i = 0 to r − 1 do
13:     w_i ← Mem w_addr + i · 8
14:     w_i ← w_i · w_i
15: end for
```

In the third step, only $||w \circ y||^2$ is left to compute according to Equation 4, where the $y$ vector from Equation 3 is element-wise multiplied by the $w$ vector. This results in a vector which then has its norm squared, expressed as a scalar product, resulting in a scalar value corresponding to the departure score

$$D = \begin{bmatrix} w_1 y_1 & w_2 y_2 & \cdots & w_r y_r \end{bmatrix} \cdot \begin{bmatrix} w_1 y_1 \\ w_2 y_2 \\ \vdots \\ w_r y_r \end{bmatrix} \quad (4)$$
$$= w_1^2 y_1^2 + w_2^2 y_2^2 + \cdots + w_r^2 y_r^2.$$

The detection phase of the PASAD algorithm has been divided into two parts. In the first part, the algorithm is initialized once by fetching data from the flash memory to the RAM memory and precomputing addresses. The second part is where the departure score calculation of the algorithm is performed.

*Initialization of the PASAD Algorithm.* The pseudocode for the initialization phase of the PASAD is presented in Algorithm 1. During the initialization phase, the $r$ and $L$ parameters are loaded into the RAM memory from the flash memory and the calculations of the addresses of the training vectors are performed. Afterwards, the $c$ vector and the $w$ vector are loaded to the RAM memory from the flash memory by using the previously calculated addresses. Additionally, the $w$ vector is element-wise multiplied by itself in order to precompute as much as possible.

**Algorithm 2** PASAD's departure score calculation algorithm

```
1:  j ← Buffer_current
2:
3:  for i = 0 to L − 1 do
4:      x_i ← Bufferj
5:      j ← j + 1  mod L
6:  end for
7:
8:  for i = 0 to r − 1 do
9:      p_i ← 0
10: end for
11:
12: DepartureScore ← 0
13: Row_addr ← U^T_addr
14:
15: for i = 0 to r − 1 do
16:     for j = 0 to L − 1 do
17:         p_i ← p_i + Mem Row_addr · x_j
18:         Row_addr ← Row_addr + 8
19:     end for
20:
21:     y ← c_i − p_i
22:     DepartureScore ← DepartureScore + w_i · y · y
23: end for
24:
25: return <DepartureScore>
```

*Departure Score Calculation of the PASAD Algorithm.* The pseudocode for the departure score calculation is presented in Algorithm 2. This part of the algorithm begins by storing the current to-be-written position of the ADC buffer in a variable $j$. Since this variable contains the to-be-written position, we know that it holds the currently oldest sampled value, which then is put first in the $x$ buffer. Next, the rest of the ADC buffer is copied to the $x$ buffer in sorted order (see lines 1 to 6). After the copying of the ADC buffer is done, the $p$ vector, which will hold the projection of $x$ onto the signal subspace $\mathcal{L}^r$, is cleared. This is performed at lines 8 to 10. Then, the departure score is cleared and the base address of the $U^T$ matrix is stored as shown at lines 12 and 13 respectively. Subsequently, the departure score is computed at lines 15 to 23. The outer loop is based on the $r$ parameter, and the reason for this is that it can perform the necessary calculations for each row in Equations 2 and 3. There is also an inner loop, based on the $L$ parameter, which calculates the $p$ vector holding the projected vector. At line 21, the projected vector is compared to the centroid vector and the result of this comparison is squared and multiplied by the corresponding element from the weight distribution vector,

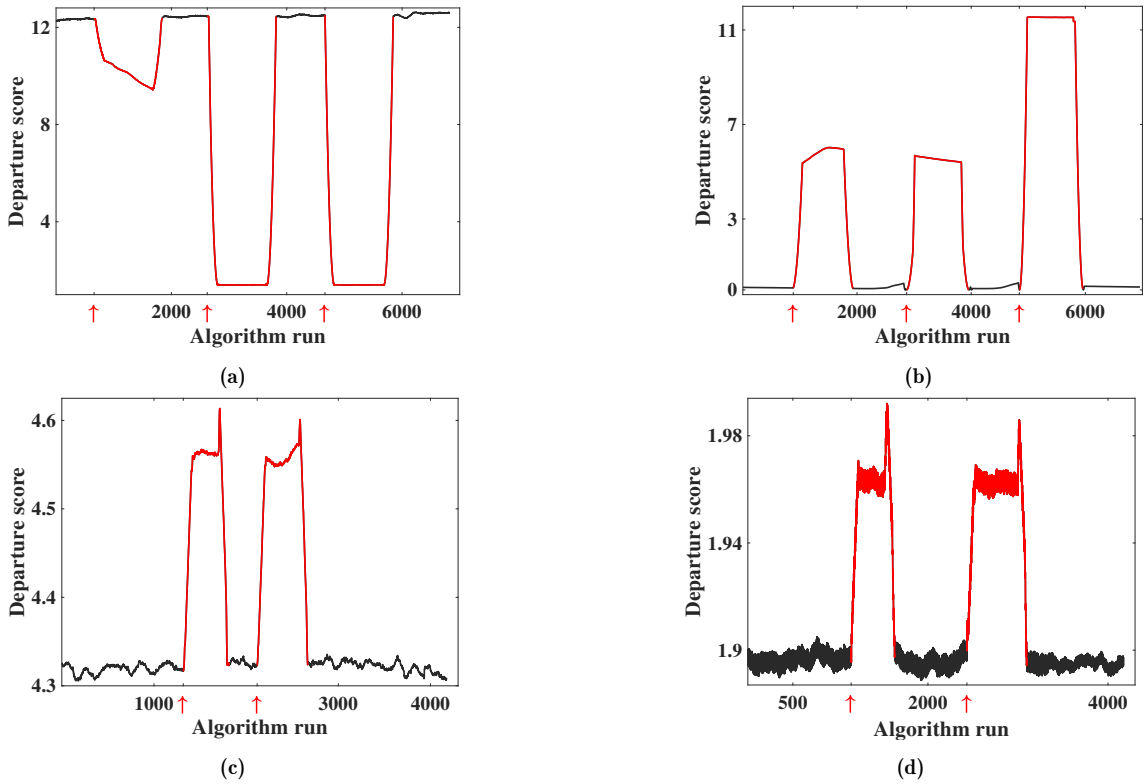



**Figure 3: Detection of several attacks (arrows on x-axis) using (a) microphone sensor (b) frequency sensor (c) load sensor without noise and (d) load sensor with noise.**

and finally added to the departure score. This is performed at line 22 and corresponds to the operations in Equation 4. Finally, at line 25, the departure score is returned.

## 6 Evaluation

This section presents the detection results of emulated attacks on sensors in a test environment as well as on real industrial machines.

### 6.1 Experiments in a Test Environment

[Exp 1] In the microphone experiments, the algorithm was trained to capture the normal behavior of a 200 Hz tone played with a specific amplitude. The three-stage attack consisted of adding a 300 Hz tone with the same amplitude, followed by only using a 300 Hz tone and, finally, by playing no sound at all. As shown in Fig. 3a, all three attacks (marked in red and whose starting point is indicated by the red arrows on the x-axis) are detectable but ill-represented, since the departure score decreased instead of increasing during the attacks.

[Exp 2] After evaluating fast-varying signals with the microphone, the frequency sensor was tested using the same attack scenarios. Since the output of the frequency sensor consists of a DC value, sampling requirements are much lower as explained in Section 4. As depicted in Fig. 3b, the frequency sensor, as opposed to the microphone sensor, was able to accurately detect all attacks, even when using different sampling rates for the training and detection phases.

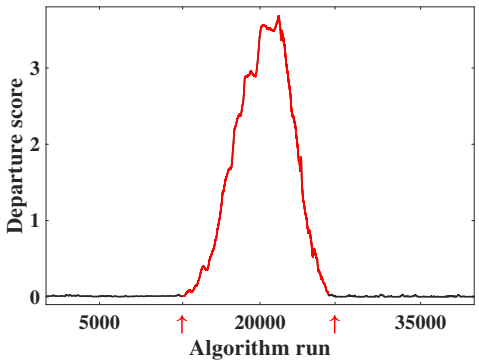
[Exp 3] In contrast to the microphone sensor, there is not as much noise that can affect the load sensor. This is due to the static behavior of the load sensor, which only changes its output when loaded with weights, and is thus not subject to interference from background sounds. The attack in this scenario consisted of adding a small weight, which was barely visible when measuring the output of the load sensor on an oscilloscope. The load sensor was evaluated both in presence and in absence of noise. To simulate noise, a spinning fan was placed on top of the sensor. The results in Fig. 3c and Fig. 3d show that all attacks in both experiments were detected.

### 6.2 Experiments on Industrial Machines

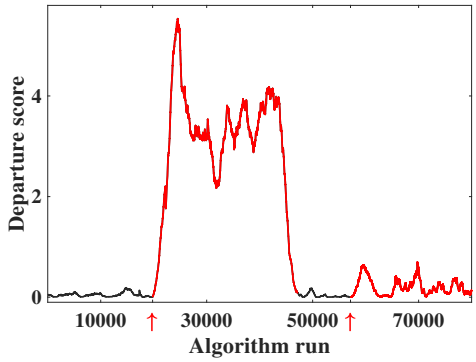
[Exp 4] Testing of the vibration sensor was performed on two types of industrial machinery: a lathe and a CNC drilling machine. The vibration sensor was fastened close to the rotating parts of the machine being tested. Then, the algorithm was trained on a specific Revolutions Per Minute (RPM). In both cases, the signal amplitudes recorded during testing were incredibly small, just above what is physically possible to detect with a 12-bit ADC. Yet, as evident in Fig. 4, even very

small changes result in a large departure score. In the lathe case, the algorithm was trained at 800 RPM, and then the lathe was attacked by stopping it, in addition to changing the RPM to 395 RPM. The results (Fig. 4a) show that the stopping of the lathe was detected, but not the change of RPM. However, it is worth mentioning that the vibrations generated from the lathe were extremely small, in addition to different frequencies interfering from various gears in the lathe's gearbox.

[Exp 5] The CNC drilling machine was running at around 500 RPM during the training phase of the algorithm. An attack was introduced by increasing the RPM to roughly 1000 RPM and then lowered to approximately 0 RPM. The results show that the algorithm was able to detect the increase of RPM, and to a lesser extent, the decrease to 0 RPM. Similarly to the lathe, the CNC drilling machine had a lot of interfering noise, even when running at 0 RPM. Also, it was difficult to adjust the RPM of the CNC drilling machine to exact RPM values.




**(a) One lathe attack detected and one missed with vibration sensor**




**(b) Detecting two different attacks on a CNC drilling machine**

**Figure 4: Detection results for experiments on real machines.**

## 7 Conclusion

Industrial control systems, which typically monitor and control critical processes, are increasingly vulnerable to cyberattacks that can cause serious damage to vital infrastructure. Process-level attack detection is a recent research trend offering techniques for detecting attacks on the physical process by monitoring low-level process data. In this work, we scrutinized the alternative side-channel based strategy of using external sensors to measure relevant physical properties of industrial machines and then monitor these measurements using existing techniques to detect malicious behavior. We designed a practical detection system that can run independently of the process flow. Moreover, we built a prototype and explained the challenges involved and possible work-arounds. Finally, we performed tests on real industrial machines, and demonstrated the viability of our approach through successful detection results.

## Acknowledgments

The research leading to these results has been supported by the Swedish Civil Contingencies Agency (MSB) through the project "RICS" and by the European Community's Horizon 2020 Framework Programme through the UNITED-GRID project under grant agreement 773717. At the time of writing this article, all authors were affiliated with the Department of Computer Science and Engineering at Chalmers University and Gothenburg University.

## References

[1] Chuadhry Mujeeb Ahmed, Jianying Zhou, and Aditya P. Mathur. 2018. Noise Matters: Using Sensor and Process Noise Fingerprint to Detect Stealthy Cyber Attacks and Authenticate Sensors in CPS. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*. ACM, New York, NY, USA, 566–581. https://doi.org/10.1145/3274694.3274748

[2] Magnus Almgren, Wissam Aoudi, Robert Gustafsson, Robin Krahl, and Andreas Lindhé. 2018. The Nuts and Bolts of Deploying Process-Level IDS in Industrial Control Systems. In *Proceedings of the 4th Annual Industrial Control System Security Workshop (ICSS '18)*. ACM, New York, NY, USA, 17–24. https://doi.org/10.1145/3295453.3295456

[3] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. 2018. Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. ACM, New York, NY, USA, 817–831. https://doi.org/10.1145/3243734.3243781

[4] J. Goh, S. Adepu, M. Tan, and Z. S. Lee. 2017. Anomaly Detection in Cyber Physical Systems Using Recurrent Neural Networks. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. 140–145. https://doi.org/10.1109/HASE.2017.36

[5] Dina Hadžiosmanović, Robin Sommer, Emmanuele Zambon, and Pieter H. Hartel. 2014. Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes. In *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC '14)*. ACM, New York, NY, USA, 126–135. https://doi.org/10.1145/2664243.2664277

[6] Karl Hoffmann. 2001. *Applying the Wheatstone Bridge Circuit*. HBM Publication. http://eln.teilam.gr/sites/default/files/Wheatstone%20bridge.pdf

[7] Marina Krotofil, Jason Larsen, and Dieter Gollmann. 2015. The Process Matters: Ensuring Data Veracity in Cyber-Physical Systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15)*. ACM, New York, NY, USA, 133–144. https://doi.org/10.1145/2714576.2714599

[8] Miro Oljaca, Peter Semig, and Collin Wells. 2015. *Connecting PGA900 Instrumentation Amplifier to Resistive Bridge Sensor*. Texas Instruments. http://www.ti.com/lit/an/slda032/slda032.pdf

[9] Dmitry Shalyga, Pavel Filonov, and Andrey Lavrentyev. 2018. Anomaly Detection for Water Treatment System based on Neural Network with Automatic Architecture Optimization. *CoRR* abs/1807.07282 (2018). arXiv:1807.07282

[10] David I. Urbina, Jairo A. Giraldo, Alvaro A. Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, New York, NY, USA, 1092–1105.

[11] David I Urbina, David I Urbina, Jairo Giraldo, Alvaro A Cardenas, Junia Valente, Mustafa Faisal, Nils Ole Tippenhauer, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. *Survey and new directions for physics-based attack detection in control systems*. National Institute of Standards and Technology.

[12] Pol Van Aubel, Kostas Papagiannopoulos, Łukasz Chmielewski, and Christian Doerr. 2017. Side-channel based intrusion detection for industrial control systems. In *International Conference on Critical Information Infrastructures Security*. Springer, 207–224.